

计算机与信息学院实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验课程： | 操作系统实验二 | | | | |
| 实验编号： |  | | | | |
| 实验名称： | 进程状态迁移 | | | | |
| 实验人员： | 年级 | | 2017级 | | |
| 专业 | | 软件工程 | | |
| 学号 | | 17111205043 | | |
| 姓名 | | 李志威 | | |
| 实验日期： | 2019.3.4 | | | | |
| 上交日期： | 2019.3.17 | | | | |
| 实 验 室： | 2号实验楼201室 | | | | |
| 实验评价： |  | | | | |
|  | 实验成绩： |  | | 评定日期： |  |
|  | 指导教师： |  | | | |

一、实验目的

写一个程序描述进程状态迁移过程，理解进程概念、状态转换及其控制

二、实验环境

操作系统：Windows10 软件：eclipse

语言：java JDK版本：10.0.1

三、实验内容与步骤

要求：

1. 提供导致进程状态变化的调用接口，包括创建、删除、调度、阻塞、时间到、挂起、激活等。
2. 实现进程列表显示的接口。
3. 这里设计的进程是一个假设的对象实体，是由程序自己创建和删除，不是系统维护的进程。

步骤：

1. 建立Process类，用于模拟进程
2. 建立PCB类，用于模拟PCB，实现PCB的各项功能
3. 编写方法，模拟PCB对进程的各种操作。通过随机数来模拟进程的阻塞与激活。当进程时间片为0时则认为进程执行完毕，进入消亡队列

源码：

public class Process {

int id;//标识符

String name;//名称

enum *State*{*READY*,*RUNNING*,*BLOCK*,*DIE*}

*State* state;//状态

int time;//需要的时间片

int priority;//优先级

/\*\*

\* 创建进程

\*/

public Process(int id, String name, int time, int priority) {

this.id = id;

this.name = name;

this.time = time;

this.priority = priority;

state = *State*.*READY*;

}

/\*\*

\* 删除进程

\*/

public void delete() {

id = 0;

name = null;

state = null;

time = 0;

priority = 0;

}

/\*\*

\* 调度进程

\*/

public void call() {

state = *State*.*RUNNING*;

time--;

priority--;

}

/\*\*

\* 阻塞进程

\*/

public void block() {

state = *State*.*BLOCK*;

priority++;

}

/\*\*

\* 时间到

\*/

public void timeOut() {

state = *State*.*DIE*;

}

/\*\*

\* 激活

\*/

public void active() {

state = *State*.*READY*;

}

}

import java.util.Queue;

import java.util.Random;

import java.util.concurrent.ArrayBlockingQueue;

public class PCB {

Queue<Process> readyQueue = new ArrayBlockingQueue(100);

Queue<Process> blockQueue = new ArrayBlockingQueue(100);

Queue<Process> dieQueue = new ArrayBlockingQueue(100);

/\*\*

\* 创建进程

\*/

public void createProcess(int id, String name, int time, int priority) {

Process process = new Process(id, name, time, priority);

addReadyQueue(process);

}

/\*\*

\* 加入就绪队列

\*/

public void addReadyQueue(Process p) {

readyQueue.add(p);

}

/\*\*

\* 从就绪队列取出

\*/

public Process removeReadyQueue() {

return readyQueue.remove();

}

/\*\*

\* 加入阻塞队列

\*/

public void addBlockQueue(Process p) {

blockQueue.add(p);

}

/\*\*

\* 从阻塞队列唤醒，并加入就绪队列

\*/

public Process removeBlockQueue() {

return blockQueue.remove();

}

/\*\*

\* 加入消亡队列

\*/

public void addDieQueue(Process p) {

dieQueue.add(p);

}

/\*\*

\* 显示各队列状态

\*/

public void printState() {

System.out.print("就绪队列：");

for (Object i : readyQueue) {

Process p = (Process) i;

System.out.print(p.name);

System.out.print(" ");

}

System.out.print("\n阻塞队列：");

for (Object i : blockQueue) {

Process p = (Process) i;

System.out.print(p.name);

System.out.print(" ");

}

System.out.print("\n死亡队列：");

for (Object i : dieQueue) {

Process p = (Process) i;

System.out.print(p.name);

System.out.print(" ");

}

}

/\*\*

\* 执行PCB

\*/

public void run() {

int flag = 1;

while (!readyQueue.isEmpty() || !blockQueue.isEmpty()) {

for (int i = 0; i < readyQueue.size(); i++) {

Process p = removeReadyQueue();

if (p.time <= 0) {

p.timeOut();

addDieQueue(p);

continue;

}

Random random = new Random();

if (random.nextBoolean()) {

addBlockQueue(p);

} else {

p.call();

addReadyQueue(p);

}

}

for (int i = 0; i < blockQueue.size(); i++) {

Process p = removeBlockQueue();

Random random = new Random();

if (random.nextBoolean()) {

p.active();

addReadyQueue(p);

} else {

p.block();

addBlockQueue(p);

}

}

System.out.println("\n第" + flag + "次运行：");

flag++;

printState();

}

System.out.println("\n所有进程运行结束");

}

}

/\*\*

\* 模拟PCB调度进程运行

\* @author 17111205043李志威

\* 2019.3.17

\*/

public class main {

public static void main(String[] args) {

PCB pcb = new PCB();

pcb.createProcess(1, "a", 1, 2);

pcb.createProcess(2, "b", 3, 4);

pcb.createProcess(3, "c", 5, 6);

pcb.createProcess(4, "d", 7, 8);

pcb.createProcess(5, "e", 9, 10);

pcb.run();

}

}

四、实验结果与分析

注：由于引用了随机数，每次运行结果不相同。

